It is claimed:

1. A method for programming an adaptive computing device, the adaptive computing device having a plurality of heterogeneous nodes coupled through a matrix interconnect network, the method comprising:

creating a first program construct having a correspondence to a selected node of the plurality of heterogeneous nodes;

creating a second program construct having a correspondence to an executable task of the selected node;

creating a third program construct having a correspondence to at least one input port coupling the selected node to the matrix interconnect network for input data to be consumed by the executable task; and

creating a fourth program construct having a correspondence to at least one output port coupling the selected node to the matrix interconnect network for output data to be produced by the executable task.

2. The method of claim 1 wherein the first program construct is a module declaration having a first unique identifier and further having a first reference to a node type corresponding to the selected node.

3. The method of claim 2 wherein the module declaration further has a second reference to one or more configuration-time parameters.

4. The method of claim 3 wherein the module declaration further has a form comprising:

[nodeType] **module** *moduleName[<parameterList>]* ,

in which *nodeType* is a placeholder for the first reference to the node type corresponding to the selected node, *moduleName* is a placeholder for the first unique identifier, and *parameterList* is a placeholder for the second reference to one or more configuration-time parameters.

5. The method of claim 2 wherein the module declaration further has a constants section which declares at least one constant which is global to the module.

6. The method of claim 2 wherein the module declaration further has a states section which declares shared state information between module processes.

7. The method of claim 6 wherein the shared state information contains an array of values stored in a memory.

8. The method of claim 2 wherein the module declaration further has a pipes section, the pipes section having the third program construct and the fourth program construct.

9. The method of claim 1 wherein the third program construct is an inpipe declaration having a first unique identifier and further having a first parameter specifying an element type of the input data and a second parameter specifying an amount of memory to be reserved for the input data; and wherein the fourth program construct is an outpipe declaration having a second unique identifier and further having a third parameter specifying an element type of the output data.

10. The method of claim 9 wherein an assignment of output data to the outpipe declaration corresponds to writing output data to the output port.

11. The method of claim 9 wherein the inpipe declaration further has a form comprising:

```
inpipe<elementType1, bufferSize> inpipeName;
```

in which *elementType1* is a placeholder for the first parameter specifying the element type of the input data, *bufferSize* is a placeholder for the second parameter specifying the amount of memory to be reserved for the input data, and *inpipeName* is a placeholder for the first unique identifier; and wherein the outpipe declaration further has a form comprising:

```
outpipe<elementType2> outpipeName;
```

in which *elementType2* is a placeholder for the third parameter specifying the element type of the output data, and *outpipeName* is a placeholder for the second unique identifier.

12. The method of claim 1 wherein the second program construct is a process declaration having a unique identifier and having at least one firing condition, the firing condition capable of determining a commencement of the executable task of the selected node.

13. The method of claim 12 wherein the process declaration further has a form comprising:

```
process processName when firingCondition {
  ...
}
```

in which *processName* is placeholder for the unique identifier, *firingCondition* is a placeholder for a condition to be fulfilled in order to commence performance of the executable task, and the ellipsis "..." is a placeholder for specification of one or more functions or algorithmic elements comprising the executable task.

14. The method of claim 1, further comprising:

providing for synchronization of production of output data with consumption of input data by:

creating a fifth program construct corresponding to a data producing task notifying a data consuming task of the creation of output data; and

creating a sixth program construct corresponding to a data consuming task notifying a data producing task of the consumption of input data.

15. The method of claim 14 wherein either the data producing task is executable on a first node of the plurality of heterogeneous nodes and the data consuming task is executable on a second node of the plurality of heterogeneous nodes or both the data producing task and the data consuming task are executable on a same node of the plurality of heterogeneous nodes.

16. The method of claim 14 wherein the fifth program construct is a notify routine and has a form comprising:

```
notify(outpipeName, numberOfElementsWritten);
```

wherein `outpipeName` is a placeholder for a first unique identifier of the fourth program construct and `numberOfElementsWritten` is a placeholder for an amount of output data produced; and wherein the sixth program construct is a release routine and has a form comprising:

```
release(inpipeName, numberOfElementsRead);
```

wherein `inpipeName` is a placeholder for a second unique identifier of the third program construct and `numberOfElementsRead` is a placeholder for an amount of input data consumed.

17. The method of claim 1, further comprising:

providing for commencement of the executable task by creating a seventh program construct having a correspondence to a task manager of the selected node.

18. The method of claim 17 wherein the seventh program construct further corresponds to an initialization of a producer count table of the task manager.

19. The method of claim 17 wherein the seventh program construct further corresponds to an initialization of a consumer count table of the task manager.

20. The method of claim 17 wherein the seventh program construct is a ready routine and has a form comprising:

```
ready(pipeName, numberOfElements);
```

wherein *pipeName* is a placeholder for a unique identifier of either the third program construct or the fourth program construct and *numberOfElements* is a placeholder for an amount of data which is sufficient for commencement of the executable task.

21. The method of claim 1, further comprising:

creating an eighth program construct linking the fourth program construct to the third program construct, the eighth program construct corresponding to a selected configuration of the matrix interconnection network providing a communication path from a selected output port to a selected input port.

22. The method of claim 21 wherein the eighth program construct is a link routine and has a form comprising:

```
link(outpipe, inpipe);
```

wherein *outpipe* is a placeholder for a first unique identifier of an instantiation of a first program construct and a fourth program construct, of a plurality of instantiations, and *inpipe* is a placeholder for a second unique identifier of an instantiation of a first program construct and a third program construct, of the plurality of instantiations.

23. The method of claim 21, further comprising:

creating a ninth program construct to instantiate a program construct of a plurality of program constructs, the plurality of program constructs comprising the first program construct, the second program construct, the third program construct, the fourth program construct, and the eighth program construct.

24. The method of claim 23 wherein the ninth program construct is a main function and has a form comprising:

```
main() {
...
}
```

wherein the ellipsis "..." is a placeholder for specification of a program construct to be instantiated.

25. The method of claim 23, wherein an instantiation corresponds to a parameter set contained within the program construct.

26. A tangible medium storing computer readable software for programming an adaptive computing device, the adaptive computing device having a plurality of heterogeneous nodes coupled through a matrix interconnect network, the tangible medium storing computer readable software comprising:

a first program construct having a correspondence to a selected node of the plurality of heterogeneous nodes;

a second program construct having a correspondence to an executable task of the selected node;

a third program construct having a correspondence to at least one input port coupling the selected node to the matrix interconnect network for input data to be consumed by the executable task; and

a fourth program construct having a correspondence to at least one output port coupling the selected node to the matrix interconnect network for output data to be produced by the executable task.

27. The tangible medium storing computer readable software of claim 26 wherein the first program construct is a module declaration having a first unique identifier and further having a first reference to a node type corresponding to the selected node.

28. The tangible medium storing computer readable software of claim 27 wherein the module declaration further has a second reference to one or more configuration-time parameters.

29. The tangible medium storing computer readable software of claim 28 wherein the module declaration further has a form comprising:

[nodeType] **module** *moduleName[<parameterList>]* ,

in which *nodeType* is a placeholder for the first reference to the node type corresponding to the selected node, *moduleName* is a placeholder for the first unique identifier, and *parameterList* is a placeholder for the second reference to one or more configuration-time parameters.

30. The tangible medium storing computer readable software of claim 27 wherein the module declaration further has a constants section which declares at least one constant which is global to the module.

31. The tangible medium storing computer readable software of claim 27 wherein the module declaration further has a states section which declares shared state information between module processes.

32. The tangible medium storing computer readable software of claim 31 wherein the shared state information contains an array of values stored in a memory.

33. The tangible medium storing computer readable software of claim 27 wherein the module declaration further has a pipes section, the pipes section having the third program construct and the fourth program construct.

34. The tangible medium storing computer readable software of claim 26 wherein the third program construct is an inpipe declaration having a first unique identifier and further having a first parameter specifying an element type of the input data and a second parameter specifying an amount of memory to be reserved for the input data; and wherein the fourth program construct is an outpipe declaration having a second unique identifier and further having a third parameter specifying an element type of the output data.

35. The tangible medium storing computer readable software of claim 34 wherein an assignment of output data to the outpipe declaration corresponds to writing output data to the output port.

36. The tangible medium storing computer readable software of claim 34 wherein the inpipe declaration further has a form comprising:

```
inpipe<elementType1, bufferSize> inpipeName;
```

in which *elementType1* is a placeholder for the first parameter specifying the element type of the input data, *bufferSize* is a placeholder for the second parameter specifying the amount of memory to be reserved for the input data, and *inpipeName* is a placeholder for the first unique identifier; and wherein the outpipe declaration further has a form comprising:

```
outpipe<elementType2> outpipeName;
```

in which *elementType2* is a placeholder for the third parameter specifying the element type of the output data, and *outpipeName* is a placeholder for the second unique identifier.

37. The tangible medium storing computer readable software of claim 26 wherein the second program construct is a process declaration having a unique identifier and having at least one firing condition, the firing condition capable of determining a commencement of the executable task of the selected node.

38. The tangible medium storing computer readable software of claim 37 wherein the process declaration further has a form comprising:

```
process processName when firingCondition {
  ...
}
```

in which *processName* is placeholder for the unique identifier, *firingCondition* is a placeholder for a condition to be fulfilled in order to commence performance of the executable task, and the ellipsis "..." is a placeholder for specification of one or more functions or algorithmic elements comprising the executable task.

39. The tangible medium storing computer readable software of claim 26, further comprising:

a fifth program construct corresponding to a data producing task notifying a data consuming task of the creation of output data; and

a sixth program construct corresponding to a data consuming task notifying a data producing task of the consumption of input data;

wherein the fifth program construct and the sixth program construct provide for synchronization of production of output data with consumption of input data.

40. The tangible medium storing computer readable software of claim 39 wherein either the data producing task is executable on a first node of the plurality of heterogeneous nodes and the data consuming task is executable on a second node of the plurality of heterogeneous nodes or both the data producing task and the data consuming task are executable on a same node of the plurality of heterogeneous nodes.

41. The tangible medium storing computer readable software of claim 39 wherein the fifth program construct is a notify routine and has a form comprising:

```
notify(outpipeName, numberOfElementsWritten);
```

wherein *outpipeName* is a placeholder for a first unique identifier of the fourth program construct and *numberOfElementsWritten* is a placeholder for an amount of output data produced; and wherein the sixth program construct is a release routine and has a form comprising:

```
release(inpipeName, numberOfElementsRead);
```

wherein *inpipeName* is a placeholder for a second unique identifier of the third program construct and *numberOfElementsRead* is a placeholder for an amount of input data consumed.

42. The tangible medium storing computer readable software of claim 26, further comprising:

a seventh program construct having a correspondence to a task manager of the selected node to provide for commencement of the executable task.

43. The tangible medium storing computer readable software of claim 42 wherein the seventh program construct further corresponds to an initialization of a producer count table of the task manager.

44. The tangible medium storing computer readable software of claim 42 wherein the seventh program construct further corresponds to an initialization of a consumer count table of the task manager.

45. The tangible medium storing computer readable software of claim 42 wherein the seventh program construct is a ready routine and has a form comprising:

```
ready(pipeName, numberOfElements);
```

wherein *pipeName* is a placeholder for a unique identifier of either the third program construct or the fourth program construct and *numberOfElements* is a placeholder for an amount of data which is sufficient for commencement of the executable task.

46. The tangible medium storing computer readable software of claim 26, further comprising:

an eighth program construct linking the fourth program construct to the third program construct, the eighth program construct corresponding to a selected configuration of the matrix interconnection network providing a communication path from a selected output port to a selected input port.

47. The tangible medium storing computer readable software of claim 46 wherein the eighth program construct is a link routine and has a form comprising:

```
link(outpipe, inpipe);
```

wherein *outpipe* is a placeholder for a first unique identifier of an instantiation of a first program construct and a fourth program construct, of a plurality of instantiations, and *inpipe* is a placeholder for a second unique identifier of an instantiation of a first program construct and a third program construct, of the plurality of instantiations.

48. The tangible medium storing computer readable software of claim 46, further comprising:

a ninth program construct to instantiate a program construct of a plurality of program constructs, the plurality of program constructs comprising the first program construct, the second program construct, the third program construct, the fourth program construct, and the eighth program construct.

49. The tangible medium storing computer readable software of claim 48 wherein the ninth program construct is a main function and has a form comprising:

```
main() {
...
}
```

wherein the ellipsis "..." is a placeholder for specification of a program construct to be instantiated.

50. The tangible medium storing computer readable software of claim 48 wherein an instantiation corresponds to a parameter set contained within the program construct.

51. A system for programming an adaptive computing device, the adaptive computing device having a plurality of heterogeneous nodes coupled through a matrix interconnect network, the system comprising:

means for a first program construct having a correspondence to a selected node of the plurality of heterogeneous nodes;

means for a second program construct having a correspondence to an executable task of the selected node, the second program construct having at least one firing condition capable of determining a commencement of the executable task of the selected node;

means for a third program construct having a correspondence to at least one input port coupling the selected node to the matrix interconnect network for input data to be consumed by the executable task;

means for a fourth program construct having a correspondence to at least one output port coupling the selected node to the matrix interconnect network for output data to be produced by the executable task;

means for a fifth program construct having a correspondence to a notification of creation of output data, and means for a sixth program construct having a correspondence to a notification of consumption of input data; wherein the fifth program construct and the sixth program construct provide for synchronization of production of output data with consumption of input data;

means for a seventh program construct having a correspondence to a task manager of the selected node to provide for commencement of the executable task, wherein the means for the seventh program construct further has correspondence to an initialization of a producer count table of the task manager or a consumer count table of the task manager; and

means for an eighth program construct linking the fourth program construct to the third program construct, the eighth program construct corresponding to a selected configuration of the matrix interconnection network providing a communication path from a selected output port to a selected input port.

52. The system of claim 51, further comprising:

means for a ninth program construct to instantiate a program construct of a plurality of program constructs, the plurality of program constructs comprising at least the first program construct, the second program construct, the third program construct, the fourth program construct, and the eighth program construct.

53. The system of claim 52, an instantiation corresponds to a parameter set contained within the program construct.